```
# Import required libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Ensemble learning models
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, GradientBoostingClassifi
```

```
# Load your diabetes dataset
df = pd.read_csv('/kaggle/input/pima-indians-diabetes-database/diabetes.csv')

# Display the first few rows of the dataset
print(df.head())
```

```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6      148             72             35        0  33.6
1            1       85             66             29        0  26.6
2            8      183             64              0        0  23.3
3            1       89             66             23       94  28.1
4            0      137             40             35      168  43.1

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1
```

```
# Check for missing values
print(df.isnull().sum())
```

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

```
# Separate features (X) and target variable (y)
X = df.drop(columns=['Outcome'])  # Features
y = df['Outcome']  # Target (1 for diabetic, 0 for non-diabetic)
```

```
# Split the data into training and test sets (70% training, 30% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

```
# Standardize the feature values
scaler = StandardScaler()
```

```
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Random Forest

```
# Initialize and train Random Forest
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

```
# Predict and evaluate performance
rf_predictions = rf_model.predict(X_test)
print("Random Forest Accuracy:", accuracy_score(y_test, rf_predictions))
print(classification_report(y_test, rf_predictions))
print(confusion_matrix(y_test, rf_predictions))
```

```
Random Forest Accuracy: 0.7575757575757576
              precision    recall  f1-score   support

           0       0.82      0.81      0.81       151
           1       0.65      0.66      0.65        80

    accuracy                           0.76       231
   macro avg       0.73      0.74      0.73       231
weighted avg       0.76      0.76      0.76       231

[[122  29]
 [ 27  53]]
```

AdaBoost

```
# Initialize and train AdaBoost
ada_model = AdaBoostClassifier(n_estimators=100, random_state=42)
ada_model.fit(X_train, y_train)
```

```
AdaBoostClassifier
AdaBoostClassifier(n_estimators=100, random_state=42)
```

```
# Predict and evaluate performance
ada_predictions = ada_model.predict(X_test)
print("AdaBoost Accuracy:", accuracy_score(y_test, ada_predictions))
print(classification_report(y_test, ada_predictions))
print(confusion_matrix(y_test, ada_predictions))
```

```
AdaBoost Accuracy: 0.70995670995671
              precision    recall  f1-score   support
```

```
              0       0.77      0.79      0.78       151
              1       0.58      0.56      0.57        80

       accuracy                           0.71       231
      macro avg       0.68      0.68      0.68       231
   weighted avg       0.71      0.71      0.71       231

   [[119  32]
    [ 35  45]]
```

Gradient Boosting

```
# Initialize and train Gradient Boosting
gb_model = GradientBoostingClassifier(n_estimators=100, random_state=42)
gb_model.fit(X_train, y_train)
```

```
▾ GradientBoostingClassifier
GradientBoostingClassifier(random_state=42)
```

```
# Predict and evaluate performance
gb_predictions = gb_model.predict(X_test)
print("Gradient Boosting Accuracy:", accuracy_score(y_test, gb_predictions))
print(classification_report(y_test, gb_predictions))
print(confusion_matrix(y_test, gb_predictions))
```

```
Gradient Boosting Accuracy: 0.7445887445887446
              precision    recall  f1-score   support

           0       0.81      0.79      0.80       151
           1       0.63      0.65      0.64        80

    accuracy                           0.74       231
   macro avg       0.72      0.72      0.72       231
weighted avg       0.75      0.74      0.75       231

[[120  31]
 [ 28  52]]
```

Compare the Performance

```
# Store and print results for easy comparison
results = {
    "Random Forest": accuracy_score(y_test, rf_predictions),
    "AdaBoost": accuracy_score(y_test, ada_predictions),
    "Gradient Boosting": accuracy_score(y_test, gb_predictions)
}

# Print all results
for model, score in results.items():
    print(f"{model}: {score:.4f}")
```

```
Random Forest: 0.7576
AdaBoost: 0.7100
Gradient Boosting: 0.7446
```